

Министерство науки и высшего образования  
Российской Федерации

Федеральное государственное бюджетное  
образовательное учреждение высшего образования  
«Новосибирский государственный технический университет»

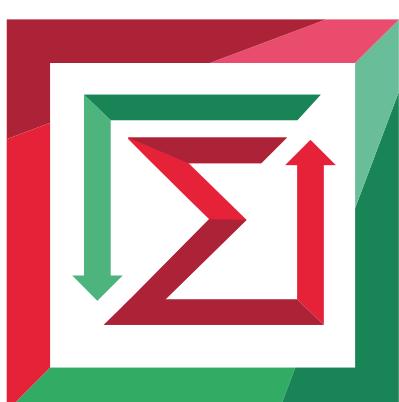


Новосибирский государственный  
технический университет  
**НЭТИ**

Кафедра теоретической и прикладной информатики

Лабораторная работа № 1  
по дисциплине «Методы оптимизации»

### МЕТОДЫ ОДНОМЕРНОГО ПОИСКА



Факультет: ПМИ  
Группа: ПМИ-81  
Бригада: 5  
Студенты: Кайда Даниил  
Парышков Дмитрий

Преподаватели: Постовалов Сергей Николаевич  
Лемешко Борис Юрьевич

Новосибирск  
2021

## 1. Цель работы

Ознакомиться с методами одномерного поиска - метод дихотомии, метод золотого сечения, метод Фибоначчи - используемыми в многомерных методах минимизации функций n переменных. Сравнить различные алгоритмы по эффективности на тестовых примерах.

## 2. Задание

Реализовать методы дихотомии, золотого сечения, Фибоначчи, исследовать их сходимость и провести сравнение по числу вычислений функции для достижения заданной точности ε от 1E-1 до 1E-7. Построить график зависимости количества вычислений минимизируемой функции от десятичного логарифма задаваемой точности ε. Реализовать алгоритм поиска интервала, содержащего минимум функции.

$$f(x) = (x - 5)^2$$

## 3. Ход работы

Реализовали методы дихотомии, золотого сечения, Фибоначчи и алгоритм поиска интервала, содержащего минимум функции.

### Общий класс для всех методов

```
class Method
{
    protected DataTable DataTable;
    protected Data Data;
    protected Function Func;
    protected int NumberOfIterationsObjectiveFunction;
    public int getNumberOfIterationsObjective()
    {
        return NumberOfIterationsObjectiveFunction;
    }
    public void ShowTable(double Eps)
    {
        DataTable.DrawTable(Eps);
    }
}
```

## Метод дихометрии

```
class DichometricsMethod : Method
{
    public DichometricsMethod()
    {
        DataTable = new DataTable();
        Func = new Function();
        Data = new Data(0, 0, 0, 0, 0, 1);
    }

    public void Do(double a, double b, double Eps = 0.001)
    {
        DataTable.ClearTable();
        double Delta = Eps / 10;
        Data.a = a;
        Data.b = b;
        Data.x1 = (Data.a + Data.b - Delta) / 2;
        Data.x2 = (Data.a + Data.b + Delta) / 2;
        Data.fx1 = Func.Value(Data.x1);
        Data.fx2 = Func.Value(Data.x2);
        DataTable.Add(Data.x1, Data.x2, Data.fx1, Data.fx2, Data.a, Data.b);

        NumberOfIterationsObjectiveFunction = 0;
        while
        (DataTable.Table[NumberOfIterationsObjectiveFunction++].difference_ab > Eps)
        {
            if (Data.fx1 < Data.fx2)
                Data.b = Data.x2;
            else
                Data.a = Data.x1;
            Data.x1 = (Data.a + Data.b - Delta) / 2;
            Data.x2 = (Data.a + Data.b + Delta) / 2;
            Data.fx1 = Func.Value(Data.x1);
            Data.fx2 = Func.Value(Data.x2);
            DataTable.Add(Data.x1, Data.x2, Data.fx1, Data.fx2, Data.a, Data.b);
        }
        NumberOfIterationsObjectiveFunction *= 2;
    }
}
```

## Метод золотого сечения

```
class GoldenSectionMethod : Method
{
    public GoldenSectionMethod()
    {
        DataTable = new DataTable();
        Func = new Function();
        Data = new Data(0, 0, 0, 0, 0, 1);
    }

    public void Do(double a, double b, double Eps = 0.001)
    {
        DataTable.ClearTable();
        Data.a = a;
        Data.b = b;
        Data.difference_ab = Math.Abs(Data.b - Data.a);
        Data.x1 = Data.a + 0.381966011 * Data.difference_ab;
        Data.x2 = Data.a + (1 - 0.381966011) * Data.difference_ab;
        Data.fx1 = Func.Value(Data.x1);
```

```

        Data.fx2 = Func.Value(Data.x2);
        DataTable.Add(Data.x1, Data.x2, Data.fx1, Data.a, Data.b);

        NumberOfIterationsObjectiveFunction = 0;
        while
(DataTable.Table[NumberOfIterationsObjectiveFunction++].difference_ab > Eps)
{
    if (Data.fx1 < Data.fx2)
    {
        Data.b = Data.x2;
        Data.x2 = Data.x1;
        Data.fx2 = Data.fx1;
        Data.difference_ab = Math.Abs(Data.b - Data.a);
        Data.x1 = Data.a + 0.381966011 * Data.difference_ab;
        Data.fx1 = Func.Value(Data.x1);
    }
    else
    {
        Data.a = Data.x1;
        Data.x1 = Data.x2;
        Data.fx1 = Data.fx2;
        Data.difference_ab = Math.Abs(Data.b - Data.a);
        Data.x2 = Data.a + (1 - 0.381966011) * Data.difference_ab;
        Data.fx2 = Func.Value(Data.x2);
    }
}

DataTable.Add(Data.x1, Data.x2, Data.fx1, Data.fx2, Data.a,
Data.b);
}
}
}

```

## Метод Фибоначчи

```

class FibonacciMethod : Method
{
    List<int> F;
    public FibonacciMethod()
    {
        DataTable = new DataTable();
        Func = new Function();
        Data = new Data(0, 0, 0, 0, 0, 1);
        F = new List<int>();
        F.Add(1);
        F.Add(1);

        const int n = 150;
        for (int i = 0; i < n; i++)
            F.Add(F[i] + F[i + 1]);
    }

    int Definition_n(double value)
    {
        int n = 0;

        while (value > F[n++ + 2]) ;

        return n;
    }

    public void Do(double a, double b, double Eps = 0.001)
    {
        DataTable.ClearTable();
        Data.a = a;
    }
}

```

```

        Data.b = b;
        Data.difference_ab = Math.Abs(Data.b - Data.a);

        int n = Definition_n(Data.difference_ab / Eps);
        int k = 0;
        double temp = F[n - k];
        double temp_2 = F[n + 2];
        double temp_3 = temp / temp_2;
        Data.x1 = Data.a + temp_3 * Data.difference_ab;
        temp = F[++n - k + 1];
        temp_3 = temp / temp_2;
        Data.x2 = Data.a + temp_3 * Data.difference_ab;

        Data.fx1 = Func.Value(Data.x1);
        Data.fx2 = Func.Value(Data.x2);
        DataTable.Add(Data.x1, Data.x2, Data.fx1, Data.fx2, Data.a, Data.b);

        NumberOfIterationsObjectiveFunction = 0;
        while
(DataTable.Table[NumberOfIterationsObjectiveFunction++].difference_ab > Eps)
{
    k++;
    if (Data.fx1 < Data.fx2)
    {
        Data.b = Data.x2;
        Data.x2 = Data.x1;
        Data.fx2 = Data.fx1;
        temp = F[n - k ];
        temp_2 = F[n + 2];
        temp_3 = temp / temp_2;
        Data.x1 = Data.a + temp_3 * Data.difference_ab;
        Data.fx1 = Func.Value(Data.x1);
    }
    else
    {
        Data.a = Data.x1;
        Data.x1 = Data.x2;
        Data.fx1 = Data.fx2;
        temp = F[n - k + 1];
        temp_3 = temp / temp_2;
        Data.x2 = Data.a + temp_3 * Data.difference_ab;
        Data.fx2 = Func.Value(Data.x2);
    }
    DataTable.Add(Data.x1, Data.x2, Data.fx1, Data.fx2, Data.a,
Data.b);
}
}

}
}

```

### **Алгоритм поиска интервала, содержащего минимум функции**

```

class IntervalSearch
{
    List<double> x;
    List<double> fx;
    Function Func;
    double a, b;

    public IntervalSearch()
    {

```

```

Func = new Function();

x = new List<double>();
fx = new List<double>();
}

public void Search(double Xzero, double Delta)
{
    x.Clear();
    fx.Clear();

    double h = Delta;

    int k = 0;

    x.Add(Xzero);
    fx.Add(Func.Value(x[k]));

    if (fx[0] > Func.Value(x[k] + Delta))
    {
        x.Add(x[k] + Delta);
        h = Delta;
    }
    else
        if ((fx[0] > Func.Value(x[k] - Delta)))
    {
        x.Add(x[k] - Delta);
        h = -Delta;
    }
    fx.Add(Func.Value(x[k + 1]));
    do
    {
        k++;
        h *= 2;
        x.Add(x[k] + h);
        fx.Add(Func.Value(x[k + 1]));
    } while (fx[k] > fx[k + 1]);

    a = x[k - 1];
    b = x[k + 1];
}

public void ShowResult()
{
    Console.WriteLine("{0,3} {1,12} {2,12}", "i", "x", "f(x)");
    for (int i = 0; i < x.Count; i++)
        Console.WriteLine("{0,3} {1: 0.0000000000} {2: 0.0000000000}",
i, x[i], fx[i]);
    Console.WriteLine("The interval containing the minimum of the
function: [{0:0.0000},{1:0.0000}]", x[x.Count - 3], x[x.Count - 1]);
}
}

```

## 4. Результаты исследований

### Метод дихотомии

Dichometrics Method:

Number Of Iterations: 58

Calculation accuracy - 1E-07

| i  | x1         | x2         | fx1                   | fx2                   | ai         | bi         | bi - ai    | (b-a)_i-1 / (b-a)_i |
|----|------------|------------|-----------------------|-----------------------|------------|------------|------------|---------------------|
| 0  | 9,0000000  | 9,0000001  | 15,999999960000000000 | 16,000000400000000000 | -2,0000000 | 20,0000000 | 22,0000000 | 1,0000000           |
| 1  | 3,5000000  | 3,5000001  | 2,2500000750000000    | 2,2499997500000000    | -2,0000000 | 9,0000001  | 11,0000001 | 0,0000000           |
| 2  | 6,2500000  | 6,2500001  | 1,56249999062500000   | 1,56250015625000000   | 3,5000000  | 9,0000001  | 5,5000001  | 3,9999999           |
| 3  | 4,8750000  | 4,8750001  | 0,015625000781250000  | 0,015624998281249800  | 3,5000000  | 6,2500001  | 2,7500001  | 3,9999999           |
| 4  | 5,5625000  | 5,5625001  | 0,316406246132812000  | 0,316406257382813000  | 4,8750000  | 6,2500001  | 1,3750001  | 3,9999998           |
| 5  | 5,21875000 | 5,21875001 | 0,047851561064453200  | 0,047851565439453600  | 4,8750000  | 5,5625001  | 0,6875001  | 3,9999996           |
| 6  | 5,04687500 | 5,04687501 | 0,002197265324707050  | 0,002197266262207170  | 4,8750000  | 5,21875001 | 0,34375001 | 3,9999991           |
| 7  | 4,96093750 | 4,96093751 | 0,001525879153442420  | 0,001525878372192390  | 4,8750000  | 5,04687501 | 0,17187501 | 3,99999983          |
| 8  | 5,00390625 | 5,00390626 | 0,000015258764190683  | 0,000015258842315726  | 4,96093750 | 5,04687501 | 0,08593751 | 3,99999965          |
| 9  | 4,98242187 | 4,98242188 | 0,00030899059009537   | 0,000308990238533044  | 4,96093750 | 5,00390626 | 0,04296876 | 3,99999930          |
| 10 | 4,99316406 | 4,99316407 | 0,000046730084962851  | 0,000046729948244126  | 4,98242187 | 5,00390626 | 0,02148438 | 3,99999860          |
| 11 | 4,99853515 | 4,99853516 | 0,00002145776531706   | 0,00002145747234865   | 4,99316406 | 5,00390626 | 0,01074220 | 3,99999721          |
| 12 | 5,00122070 | 5,00122071 | 0,00001490108349928   | 0,00001490132764029   | 4,99853515 | 5,00390626 | 0,00537110 | 3,99999441          |
| 13 | 4,99987793 | 4,99987794 | 0,00000014901938001   | 0,00000014899496631   | 4,99853515 | 5,00122071 | 0,00268556 | 3,99998883          |
| 14 | 5,00054931 | 5,00054932 | 0,00000301745018261   | 0,00000301756004627   | 4,99987793 | 5,00122071 | 0,00134278 | 3,99997766          |
| 15 | 5,00021362 | 5,00021363 | 0,00000045633446705   | 0,00000045637719203   | 4,99987793 | 5,00054932 | 0,00067140 | 3,99995532          |
| 16 | 5,00004577 | 5,00004578 | 0,00000002095184497   | 0,00000002096100061   | 4,99987793 | 5,00021363 | 0,00033570 | 3,99991064          |
| 17 | 4,99996185 | 4,99996186 | 0,00000001455434285   | 0,00000001454671382   | 4,99987793 | 5,00004578 | 0,00016786 | 3,99982128          |
| 18 | 5,00000381 | 5,00000382 | 0,00000000014527650   | 0,00000000014603980   | 4,99996185 | 5,00004578 | 0,00008393 | 3,99964257          |
| 19 | 4,99998283 | 4,99998284 | 0,000000000294785532  | 0,00000000029442246   | 4,99996185 | 5,00000382 | 0,00004197 | 3,99928523          |
| 20 | 4,99999332 | 4,99999333 | 0,00000000044607732   | 0,00000000044474254   | 4,99998283 | 5,00000382 | 0,00002099 | 3,99857080          |
| 21 | 4,99999857 | 4,99999858 | 0,0000000002055476    | 0,0000000002026903    | 4,99999332 | 5,00000382 | 0,00001050 | 3,99714297          |
| 22 | 5,00000119 | 5,00000120 | 0,0000000001413510    | 0,0000000001437388    | 4,99999857 | 5,00000382 | 0,00000526 | 3,99429138          |
| 23 | 4,99999988 | 4,99999989 | 0,0000000000014980    | 0,0000000000012632    | 4,99999857 | 5,00000120 | 0,00000263 | 3,98860444          |
| 24 | 5,00000053 | 5,00000054 | 0,00000000000284366   | 0,00000000000295131   | 4,99999988 | 5,00000120 | 0,00000132 | 3,97729513          |
| 25 | 5,00000021 | 5,00000022 | 0,00000000000042203   | 0,00000000000046412   | 4,99999988 | 5,00000054 | 0,00000067 | 3,95493134          |
| 26 | 5,00000004 | 5,00000005 | 0,00000000000001724   | 0,00000000000002654   | 4,99999988 | 5,00000022 | 0,00000034 | 3,91119676          |
| 27 | 4,99999996 | 4,99999997 | 0,00000000000001635   | 0,00000000000000926   | 4,99999988 | 5,00000005 | 0,00000017 | 3,82749972          |
| 28 | 5,00000000 | 5,00000001 | 0,000000000000000000  | 0,0000000000000111    | 4,99999996 | 5,00000005 | 0,00000009 | 3,67375836          |

Result: x = 5,000000005543473

## Метод золотого сечения

GoldenSection Method:

Number Of Iterations: 41

Calculation accuracy - 1E-07

| i  | x1         | x2          | fx1                   | fx2                   | ai          | bi          | bi - ai     | (b-a)_i-1 / (b-a)_i |
|----|------------|-------------|-----------------------|-----------------------|-------------|-------------|-------------|---------------------|
| 0  | 6,40325224 | 11,59674776 | 1,969116854678030000  | 43,517080982678000000 | -2,00000000 | 20,00000000 | 22,00000000 | 1,00000000          |
| 1  | 3,19349550 | 6,40325224  | 3,263458495164930000  | 1,969116854678030000  | -2,00000000 | 11,59674776 | 13,59674776 | 0,00000000          |
| 2  | 6,40325224 | 8,38699101  | 1,969116854678030000  | 11,471708135669800000 | 3,19349550  | 11,59674776 | 8,40325225  | 2,61803399          |
| 3  | 5,17723427 | 6,40325224  | 0,031411985503346100  | 1,969116854678030000  | 3,19349550  | 8,38699101  | 5,19349551  | 2,61803399          |
| 4  | 4,41951348 | 5,17723427  | 0,336964598384748000  | 0,031411985503346100  | 3,19349550  | 6,40325224  | 3,20975674  | 2,61803399          |
| 5  | 5,17723427 | 5,64553146  | 0,031411985503346100  | 0,416710866768801000  | 4,41951348  | 6,40325224  | 1,98373876  | 2,61803399          |
| 6  | 4,88781068 | 5,17723427  | 0,012586443900290700  | 0,031411985503346100  | 4,41951348  | 5,64553146  | 1,22601798  | 2,61803399          |
| 7  | 4,70893707 | 4,88781068  | 0,084717630744118500  | 0,012586443900290700  | 4,41951348  | 5,17723427  | 0,75772079  | 2,61803397          |
| 8  | 4,88781068 | 4,99836065  | 0,012586443900290700  | 0,000002687455696655  | 4,70893707  | 5,17723427  | 0,46829720  | 2,61803397          |
| 9  | 4,99836065 | 5,06668429  | 0,000002687455696655  | 0,004446795002570300  | 4,88781068  | 5,17723427  | 0,28942359  | 2,61803396          |
| 10 | 4,95613432 | 4,99836065  | 0,001924197917966180  | 0,000002687455696655  | 4,88781068  | 5,06668429  | 0,17887362  | 2,61803396          |
| 11 | 4,99836065 | 5,02445796  | 0,000002687455696655  | 0,000598191854450591  | 4,95613432  | 5,06668429  | 0,11054997  | 2,61803399          |
| 12 | 4,98223163 | 4,99836065  | 0,000315715031406571  | 0,000002687455696655  | 4,95613432  | 5,02445796  | 0,06832364  | 2,61803399          |
| 13 | 4,99836065 | 5,00832894  | 0,000002687455696655  | 0,000069371193749846  | 4,98223163  | 5,02445796  | 0,04222633  | 2,61803399          |
| 14 | 4,99219991 | 4,99836065  | 0,000060841352872935  | 0,000002687455696655  | 4,98223163  | 5,00832894  | 0,02609731  | 2,61803399          |
| 15 | 4,99836065 | 5,00216820  | 0,000002687455696655  | 0,000004701083560880  | 4,99219991  | 5,00832894  | 0,01612902  | 2,61803399          |
| 16 | 4,99600746 | 4,99836065  | 0,0000015940381103707 | 0,000002687455696655  | 4,99219991  | 5,00216820  | 0,00996828  | 2,61803399          |
| 17 | 4,99836065 | 4,99981501  | 0,000002687455696655  | 0,000000034223016036  | 4,99600746  | 5,00216820  | 0,00616074  | 2,61803399          |
| 18 | 4,99981501 | 5,00071385  | 0,000000034223016036  | 0,000000509575687582  | 4,99836065  | 5,00216820  | 0,00380754  | 2,61803515          |
| 19 | 4,99925949 | 4,99981501  | 0,000000548350358510  | 0,000000034223016036  | 4,99836065  | 5,00071385  | 0,00235319  | 2,61803515          |
| 20 | 4,99981501 | 5,00015833  | 0,000000034223016036  | 0,000000025069170765  | 4,99925949  | 5,00071385  | 0,00145435  | 2,61803399          |
| 21 | 5,00015833 | 5,00037052  | 0,000000025069170765  | 0,000000137284509752  | 4,99981501  | 5,00071385  | 0,00089884  | 2,61803094          |
| 22 | 5,00002719 | 5,00015833  | 0,00000000739447471   | 0,000000025069170765  | 4,99981501  | 5,00037052  | 0,00055551  | 2,61803094          |
| 23 | 4,99994614 | 5,00002719  | 0,000000002900398977  | 0,00000000739447471   | 4,99981501  | 5,00015833  | 0,00034333  | 2,61802905          |
| 24 | 5,00002719 | 5,00007728  | 0,000000000739447471  | 0,000000005972806353  | 4,99994614  | 5,00015833  | 0,00021219  | 2,61802905          |
| 25 | 4,99999624 | 5,00002719  | 0,000000000014172232  | 0,00000000739447471   | 4,99994614  | 5,00007728  | 0,00013114  | 2,61803399          |
| 26 | 4,99997710 | 4,99999624  | 0,000000000524305546  | 0,000000000014172232  | 4,99994614  | 5,00002719  | 0,00008105  | 2,61804691          |
| 27 | 4,99999624 | 5,00000806  | 0,000000000014172232  | 0,00000000064962213   | 4,99997710  | 5,00002719  | 0,00005009  | 2,61804691          |
| 28 | 4,99998893 | 4,99999624  | 0,000000000122610345  | 0,000000000014172232  | 4,99997710  | 5,00000806  | 0,00003096  | 2,61803399          |
| 29 | 4,99999624 | 5,00000075  | 0,000000000014172232  | 0,00000000000565215   | 4,99998893  | 5,00000806  | 0,00001913  | 2,61803399          |
| 30 | 5,00000075 | 5,00000354  | 0,00000000000565215   | 0,000000000012555333  | 4,99999624  | 5,00000806  | 0,00001182  | 2,61808872          |
| 31 | 4,99999903 | 5,00000075  | 0,00000000000947143   | 0,00000000000565215   | 4,99999624  | 5,00000354  | 0,00000731  | 2,61808872          |
| 32 | 5,00000075 | 5,00000182  | 0,00000000000565215   | 0,000000000003305767  | 4,99999903  | 5,00000354  | 0,00000452  | 2,61803399          |
| 33 | 5,00000009 | 5,00000075  | 0,0000000000008650    | 0,00000000000565215   | 4,99999903  | 5,00000182  | 0,00000279  | 2,61803399          |
| 34 | 4,99999969 | 5,00000009  | 0,0000000000098793    | 0,0000000000008650    | 4,99999903  | 5,00000075  | 0,00000173  | 2,61826586          |
| 35 | 5,00000009 | 5,00000034  | 0,0000000000008650    | 0,00000000000118739   | 4,99999969  | 5,00000075  | 0,00000107  | 2,61826586          |
| 36 | 4,99999994 | 5,00000009  | 0,0000000000003923    | 0,0000000000008650    | 4,99999969  | 5,00000034  | 0,00000066  | 2,61803399          |
| 37 | 4,99999994 | 4,99999994  | 0,00000000000025196   | 0,0000000000003923    | 4,99999969  | 5,00000009  | 0,00000041  | 2,61742708          |
| 38 | 4,99999994 | 5,00000000  | 0,00000000000003923   | 0,000000000000010     | 4,99999984  | 5,00000009  | 0,00000025  | 2,61742708          |

## Метод Фибоначчи

Fibonacci Method:

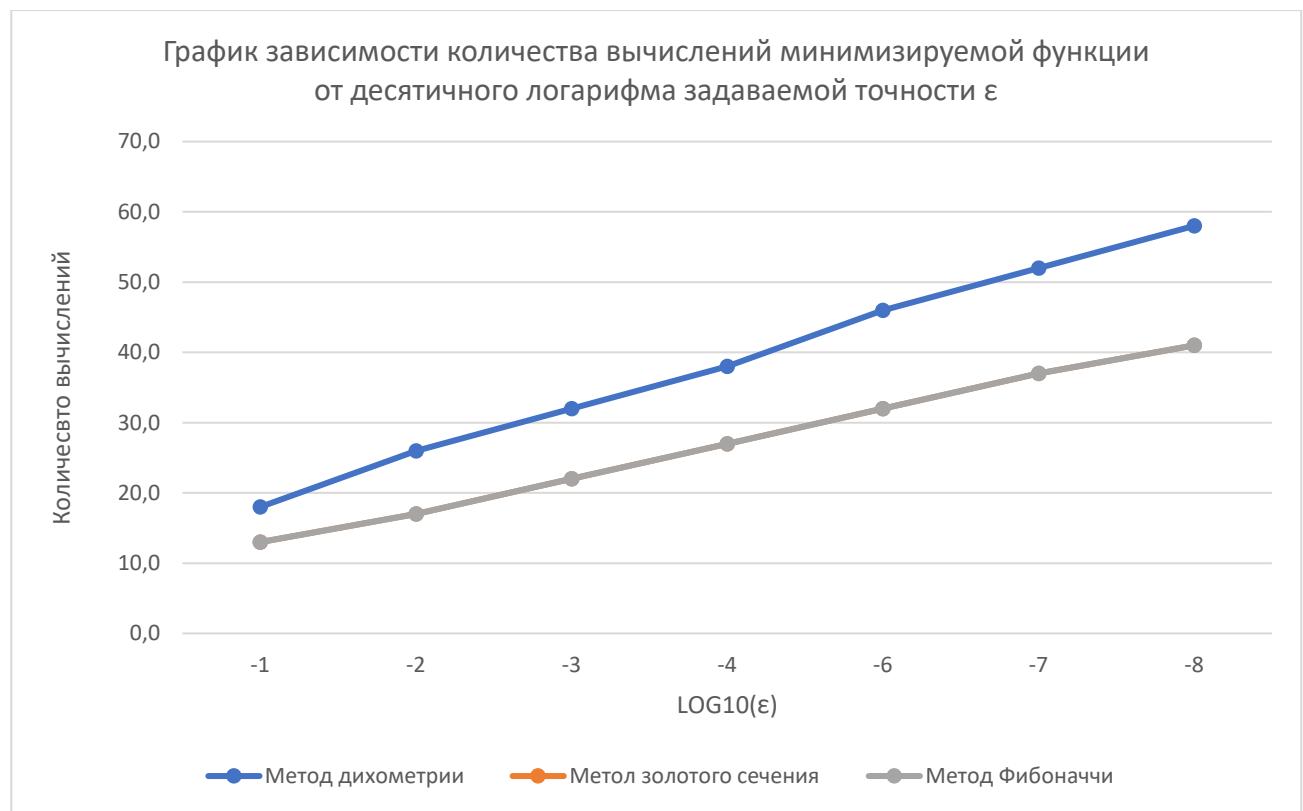
Number Of Iterations: 41

Calculation accuracy - 1E-07

| i  | x1         | x2          | fx1                  | fx2                        | ai          | bi          | bi - ai     | (b-a)_i-1 / (b-a)_i |
|----|------------|-------------|----------------------|----------------------------|-------------|-------------|-------------|---------------------|
| 0  | 6,40325225 | 20,00000000 | 1,969116870120300000 | 225,0000000000000000000000 | -2,00000000 | 20,00000000 | 22,00000000 | 1,00000000          |
| 1  | 3,19349550 | 6,40325225  | 3,263458490471920000 | 1,969116870120300000       | -2,00000000 | 20,00000000 | 22,00000000 | 0,00000000          |
| 2  | 6,40325225 | 8,38699101  | 1,969116870120300000 | 11,471708101758100000      | 3,19349550  | 20,00000000 | 16,80650450 | 1,30901699          |
| 3  | 5,17723427 | 6,40325225  | 0,031411985570521300 | 1,969116870120300000       | 3,19349550  | 8,38699101  | 5,19349550  | 4,23606798          |
| 4  | 4,41951349 | 5,17723427  | 0,336964594080731000 | 0,031411985570521300       | 3,19349550  | 6,40325225  | 3,20975674  | 5,23606798          |
| 5  | 5,17723427 | 5,64553147  | 0,031411985570521300 | 0,416710872346860000       | 4,41951349  | 6,40325225  | 1,98373876  | 2,61803399          |
| 6  | 4,88781068 | 5,17723427  | 0,012586442947101100 | 0,031411985570521300       | 4,41951349  | 5,64553147  | 1,22601798  | 2,61803399          |
| 7  | 4,70893707 | 4,88781068  | 0,084717629257890900 | 0,012586442947101100       | 4,41951349  | 5,17723427  | 0,75772078  | 2,61803399          |
| 8  | 4,88781068 | 4,99836065  | 0,012586442947101100 | 0,000002687452499274       | 4,70893707  | 5,17723427  | 0,46829720  | 2,61803399          |
| 9  | 4,99836065 | 5,06668430  | 0,000002687452499274 | 0,004446795224946070       | 4,88781068  | 5,17723427  | 0,28942358  | 2,61803399          |
| 10 | 4,95613432 | 4,99836065  | 0,001924197627829380 | 0,000002687452499274       | 4,88781068  | 5,06668430  | 0,17887361  | 2,61803399          |
| 11 | 4,99836065 | 5,02445796  | 0,000002687452499274 | 0,000598191965296283       | 4,95613432  | 5,06668430  | 0,11054997  | 2,61803399          |
| 12 | 4,98223163 | 4,99836065  | 0,000315714927406977 | 0,000002687452499274       | 4,95613432  | 5,02445796  | 0,06832364  | 2,61803399          |
| 13 | 4,99836065 | 5,00832894  | 0,000002687452499274 | 0,000069371235523792       | 4,98223163  | 5,02445796  | 0,04222633  | 2,61803399          |
| 14 | 4,99219992 | 4,99836065  | 0,000060841309612282 | 0,000002687452499274       | 4,98223163  | 5,00832894  | 0,02609731  | 2,61803399          |
| 15 | 4,99836065 | 5,00216820  | 0,000002687452499274 | 0,000004701094857395       | 4,99219992  | 5,00832894  | 0,01612902  | 2,61803399          |
| 16 | 4,99600746 | 4,99836065  | 0,000015940359453355 | 0,000002687452499274       | 4,99219992  | 5,00216820  | 0,00996828  | 2,61803399          |
| 17 | 4,99836065 | 4,99981501  | 0,000002687452499274 | 0,00000034222037777        | 4,99600746  | 5,00216820  | 0,00616074  | 2,61803399          |
| 18 | 4,99981501 | 5,00071385  | 0,000000034222037777 | 0,000000509578516789       | 4,99836065  | 5,00216820  | 0,00380755  | 2,61803399          |
| 19 | 4,99925949 | 4,99981501  | 0,000000548348343726 | 0,000000034222037777       | 4,99836065  | 5,00071385  | 0,00235319  | 2,61803399          |
| 20 | 4,99981501 | 5,00015833  | 0,000000034222037777 | 0,000000025069723128       | 4,99925949  | 5,00071385  | 0,00145435  | 2,61803399          |
| 21 | 5,00015833 | 5,00037052  | 0,000000025069723128 | 0,000000137286165203       | 4,99981501  | 5,00071385  | 0,00089884  | 2,61803399          |
| 22 | 5,00002720 | 5,00015833  | 0,00000000739582720  | 0,000000025069723128       | 4,99981501  | 5,00037052  | 0,00055551  | 2,61803399          |
| 23 | 4,99994615 | 5,00002720  | 0,000000002900151060 | 0,00000000739582720        | 4,99981501  | 5,00015833  | 0,00034333  | 2,61803399          |
| 24 | 5,00002720 | 5,00007729  | 0,00000000739582720  | 0,000000005973109194       | 4,99994615  | 5,00015833  | 0,00021219  | 2,61803400          |
| 25 | 4,99999624 | 5,00002720  | 0,00000000014155866  | 0,00000000739582720        | 4,99994615  | 5,00007729  | 0,00013114  | 2,61803396          |
| 26 | 4,99997710 | 4,99999624  | 0,000000000524197154 | 0,00000000014155866        | 4,99994615  | 5,00002720  | 0,00008105  | 2,61803406          |
| 27 | 4,99999624 | 5,00000806  | 0,00000000014155866  | 0,00000000065001425        | 4,99997710  | 5,00002720  | 0,00005009  | 2,61803381          |
| 28 | 4,99998893 | 4,99999624  | 0,000000000122557694 | 0,00000000014155866        | 4,99997710  | 5,00000806  | 0,00003096  | 2,61803445          |
| 29 | 4,99999624 | 5,00000075  | 0,00000000014155866  | 0,0000000000568812         | 4,99998893  | 5,00000806  | 0,00001913  | 2,61803279          |
| 30 | 5,00000075 | 5,0000355   | 0,0000000000568812   | 0,00000000012572141        | 4,99999624  | 5,00000806  | 0,00001182  | 2,61803714          |
| 31 | 4,99999903 | 5,00000075  | 0,0000000000942653   | 0,0000000000568812         | 4,99999624  | 5,00000355  | 0,00000731  | 2,61802575          |
| 32 | 5,00000075 | 5,00000182  | 0,0000000000568812   | 0,00000000003314664        | 4,99999903  | 5,00000355  | 0,00000452  | 2,61805556          |
| 33 | 5,00000010 | 5,00000075  | 0,000000000009124    | 0,0000000000568812         | 4,99999903  | 5,00000182  | 0,00000279  | 2,61797753          |
| 34 | 4,99999969 | 5,00000010  | 0,000000000097487    | 0,000000000009124          | 4,99999903  | 5,00000075  | 0,00000173  | 2,61818182          |
| 35 | 5,00000010 | 5,00000035  | 0,000000000009124    | 0,000000000012025          | 4,99999969  | 5,00000075  | 0,00000107  | 2,61764706          |
| 36 | 4,99999994 | 5,00000010  | 0,000000000003758    | 0,000000000009124          | 4,99999969  | 5,00000035  | 0,00000066  | 2,61904762          |
| 37 | 4,99999984 | 4,99999994  | 0,0000000000024150   | 0,000000000003758          | 4,99999969  | 5,00000010  | 0,00000041  | 2,61538462          |
| 38 | 4,99999994 | 5,00000000  | 0,000000000003758    | 0,00000000000002           | 4,99999984  | 5,00000010  | 0,00000025  | 2,62500001          |
| 39 | 5,00000000 | 5,00000003  | 0,000000000000002    | 0,000000000001075          | 4,99999994  | 5,00000010  | 0,00000016  | 2,60000001          |
| 40 | 4,99999997 | 5,00000000  | 0,000000000000896    | 0,00000000000002           | 4,99999994  | 5,00000003  | 0,00000009  | 2,66666666          |

## Количество вычислений целевой функции для каждого метода в зависимости от заданной точности

| Number Of Iterations Objective Function: |     |     |      |       |        |       |       |       |
|--|-----|-----|------|-------|--------|-------|-------|-------|
| Name Method                              | Eps | 0,1 | 0,01 | 0,001 | 0,0001 | 1E-05 | 1E-06 | 1E-07 |
| Dichometrics                             |     | 18  | 26   | 32    | 38     | 46    | 52    | 58    |
| Golden Section                           |     | 13  | 17   | 22    | 27     | 32    | 37    | 41    |
| Fibonacci                                |     | 13  | 17   | 22    | 27     | 32    | 37    | 41    |



### Вывод

График показывает линейную зависимость между количеством вычислений минимизируемой функции от логарифма задаваемой точности  $\varepsilon$ , следовательно, чем точнее необходимо решение, тем большее количество итераций необходимо сделать.

Количество вычислений минимизируемой функции для равной точности у методов золотого сечения и Фибоначчи меньше, чем у метода дихотомии, но количество итераций у метода дихотомии меньше. Методы золотого сечения и Фибоначчи эффективно использовать для функций, вычисление которых затратно по времени или/и ресурсам.

Метод золотого сечения и метод Фибоначчи обладают приблизительно одинаковой скоростью сходимости, лучшей, чем у метода дихотомии. Метод Фибоначчи также позволяет заранее предсказать количество итераций ценой большей вычислительной

сложности, но при этом необходимо вычислять значение функции Фибоначчи, что несёт дополнительные затраты по вычислительной мощности.

**Таблица, показывающая процесс поиска интервала, содержащего минимум:**

$$\delta = 0.1, x_0 = 4$$

| $i$ | $x_i$ | $f(x_i)$ |
|-----|-------|----------|
| 0   | 0     | 25.00    |
| 1   | 0.1   | 24.01    |
| 2   | 0.3   | 22.09    |
| 3   | 0.7   | 18.49    |
| 4   | 1.5   | 12.25    |
| 5   | 3.1   | 3.61     |
| 6   | 6.3   | 1.69     |
| 7   | 12.7  | 59.29    |

**Интервал, содержащий минимум: [3.1, 12.7]**

#### **Вывод**

При приближении начального приближения к точке минимума количество итераций уменьшается. Как только точка минимума была пройдена, итерационный процесс заканчивается.

